Introduction to Geant4

CSC2000 - Marathon

Makoto ASAI
Hiroshima Institute of Technology
(Geant4 / ATLAS)
Makoto.Asai@cern.ch

Contents

- PART 0 - What is Geant4?
  - What is Geant4?
  - Its history and future
  - Geant4 collaboration
  - Performance? Flexibility? Usability?
- PART 1 - Jump into Geant4 world
  - Basic concepts in Geant4
  - How Geant4 runs
  - Global structure of Geant4 toolkit

2

Contents

- PART 2 - The minimal things you have to do
  - The main program
  - Describe your detector
  - Select physics processes
  - Generate primary event
  - Environment variables
- PART 3 - Add optional features
  - Select (G)UI
  - Visualization
  - Optional user action classes

3

Contents

- PART 4 - Basics about geometry
  - Unit system
  - Material
  - Define detector geometry
  - Magnetic field
  - Touchable
- PART 5 - Sensitive detector and digitizer
  - Detector sensitivity
  - Hit class
  - Readout geometry
  - Digitization

4

Contents

- PART 6 - Visualization and (G)UI
  - Visualization of Detector
  - Visualization of Hits and Trajectories
  - What is Intercoms?
  - Define user commands
  - G4cout / G4cerr / G4endl
- PART 7 - Learn more
  - User's manuals
  - Examples

5

PART 0

What is Geant4?

What is Geant4?

- Geant4 is the successor of GEANT3, the world-standard toolkit for HEP detector simulation.
- Geant4 is one of the first successful attempt to re-design a major package of CERN software for the next generation of HEP experiments using an Object-Oriented environment.
- A variety of requirements also came from heavy ion physics, CP violation physics, cosmic ray physics, medical applications and space science applications.
- In order to meet such requirements, a large degree of functionality and flexibility are provided.
- G4 is not only for HEP but goes well beyond that.

7

Geant4 - Its history and future

- Limitations of GEANT3 maintenance
  - Because of too complex structure driven by too many historical reasons, it became impossible to add a new feature or to hunt a bug.

    ---> Limitation of FORTRAN
  - Shortage of man power at CERN

    ---> Limitation of “central center” supports
- World-wide collaboration
  - Adoption of the most recent software engineering methodologies
  - Choice of Object-orientation and C++

8

Geant4 - Its history and future

- Dec ’94 - Project start
- Apr ’97 - First alpha release
- Jul ’98 - First beta release
- Dec ’98 - Geant4 0.0 release
- Jul ’99 - Geant4 0.1 release
- Jun ’00 - Geant4 2.0 release
- We will continue to maintain and upgrade Geant4 for at least 10 years.

CERN RD44

MoU-based collaboration

9

Performance?

- We believe that Geant4 is a fundamental test of the suitability of the object-oriented approach for software in HEP, where performance is an important issue.
- As a consequence, Geant4 releases should be regularly monitored against the performance provided by GEANT3 at comparable physics accuracy.

10

Performance?

- Geometry navigation
  - Geant4 automatically optimizes the user's geometrical description. And it provides faster navigation than optimized Geant3 descriptions.
- EM Physics in a simple sampling calorimeter
  - 3 times faster when using the same cuts (in the sensitive material) as GEANT3.
  - More than a factor 10 faster when seeking the best performance in Geant4 that maintains constant the quality of the physics results.
- Geant4 is faster than GEANT3 in all aspects.
  - when its power and features are well exploited.

11

Flexibility?

- Much wider coverage of physics comes from mixture of theory-driven, cross-section tables, and empirical formulae. Thanks to polymorphism mechanism, both cross-sections and models can be combined in arbitrary manners into one particular process.
  - Slow neutron
  - Ultra-high energy muon
  - Optical photon
  - Parton string models
  - Shower parameterization
  - Event biasing technique
  - new areas are coming...

12

Flexibility?

- Many types of geometrical descriptions
  - CSG, BREP, Boolean
  - STEP compliant
- Event and Track are class objects
  - Overlap events
  - Suspend slow looping tracks and postpone them to next event
  - Priority control of tracks without performance overhead
- Everything is open to the user
  - Choice of physics processes / models
  - Choice of GUI / Visualization technology

13

Usability?

- User Requirements Document states many different use-cases from various fields.
- Thanks to the inheritance mechanism, the user can derive his/her own classes easily. Many abstract layers and default behaviors are provided at the same time.
- Many reusable examples and documents are provided and are still continuously evolving with the user's contribution.

14

Comparison projects

- We are establishing projects for comparing G4 results with experimental data and/or test beam data.
- Projects are planned to get first results by end of this year with publications.
- Projects are achieved by close collaborations with experiments
  - ATLAS
  - BaBar
  - CMS
  - ESA

15

PART 1

Jump into Geant4 world

Detector simulation standing on Object-Orientation

- Simulation in HEP is a "virtual reality". Simulation is used both to help designing detectors during R&D phase and understanding the response of the detector for the physics studies.
- To create such virtual reality we need to model the particle-matter interactions, geometry and materials in order to propagate elementary particles into the detector.
- We need also to describe the sensitivity of the detector for generating raw data.

17

Detector simulation standing on Object-Orientation

- Geant4 is the Object-Oriented toolkit which provides functionalities required for simulations in HEP and other fields.
- Benefits of Object-Orientation help you to realize a detector simulator which is
  - Easy to develop and maintain
  - Well modularized
  - Readable and Understandable to the collaborators

18

Basic concepts in Geant4

- Run, Event, Track, Step, Trajectory
- Physics process and cut-off
- Sensitive detector and Hit
- Manager classes

19

Run

- As an analogy of the real experiment, a run of Geant4 starts with “Beam On”.
- Within a run, the user cannot change
  - detector geometry
  - settings of physics processes

  ---> detector is inaccessible during a run
- Conceptually, a run is a collection of events which share the same detector conditions.

20

Event

- At beginning of processing, an event contains primary particles. These primaries are pushed into a stack.
- When the stack becomes empty, processing of an event is over.
- G4Event class represents an event. It has following objects at the end of its processing.
  - List of primary vertexes and particles
  - Trajectory collection (optional)
  - Hits collections
  - Digits collections (optional)

21

Track

- Track is a snapshot of a particle.
- Step is a “delta” information to a track.
  - Track is not a collection of steps.
- Track is deleted when
  - it goes out of the world volume
  - it disappears (e.g. decay)
  - it goes down to zero kinetic energy and no “at rest” additional process is required
  - the user decides to kill it

22

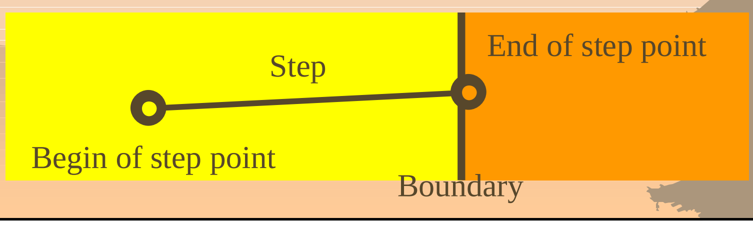
Track

- A track is made of three layers of class objects.
  - G4Track
    - Position, volume, track length, global ToF
    - ID of itself and mother track
  - G4DynamicParticle
    - Momentum, energy, local time, polarization
    - Pre-fixed decay channel
  - G4ParticleDefinition
    - Shared by all G4DynamicParticle of same type
    - Mass, lifetime, charge, other physical quantities
    - Decay table

23

Step

- Step has two points and also “delta” information of a particle (energy loss on the step, time-of-flight spent by the step, etc.).
- Each point knows the volume. In case a step is limited by a volume boundary, the end point physically stands on the boundary, and it logically belongs to the next volume.



24

Trajectory

- Trajectory is a record of a track history. It stores some information of all steps done by the track as objects of G4TrajectoryPoint class.
- It is advised not to store trajectories for secondary particles generated in a shower because of the memory consumption.
- The user can create his own trajectory class deriving from G4VTrajectory and G4VTrajectoryPoint base classes for storing any additional information useful to the simulation.

25

Physics process

- Three basic types
  - At rest process (e.g. decay at rest)
  - Continuous process (e.g. ionization)
  - Discrete process (e.g. decay on the fly)
- Transportation is still a process.
  - Interacting with volume boundary
  - Parameterization can take over
- A process which requires the shortest physical interaction length limits the step.
- A “Logical Volume” can have its own “user limits”.

26

Cut-off

- In Geant4, the user defines cut-off by length instead of energy.
  - It makes poor sense to use the energy cut-off.
    - Range of 10 keV gamma in Si ~ a few cm
    - Range of 10 keV electron in Si ~ a few micron
  - Cut-off represents the accuracy of the stopping position. It does not mean that the track is killed at the corresponding energy.
  - In Geant4, a track reached to the cut-off is traced down to zero kinetic energy with one additional step. Additional "AtRest" process may occur.

27

Cut-off

- In case the energy corresponding to the given cut-off in a thin material is less than the available energy range of a physics process, Geant4 will not stop that particle by that process in the current volume (material).
  - In case the track goes into another volume (material) which is more dense, that process may stop the track.

28

Sensitive detector and Hit

- Each "Logical Volume" can have a pointer to a sensitive detector.
- Hit is a snapshot of the physical interaction of a track or an accumulation of interactions of tracks in the sensitive region of your detector.
- A sensitive detector creates hit(s) using the information given in G4Step object. The user has to provide his/her own implementation of the detector response.
- Hit objects, which still are the user's class objects, are collected in a G4Event object at the end of an event.
  - UserSteppingAction class should not do this.

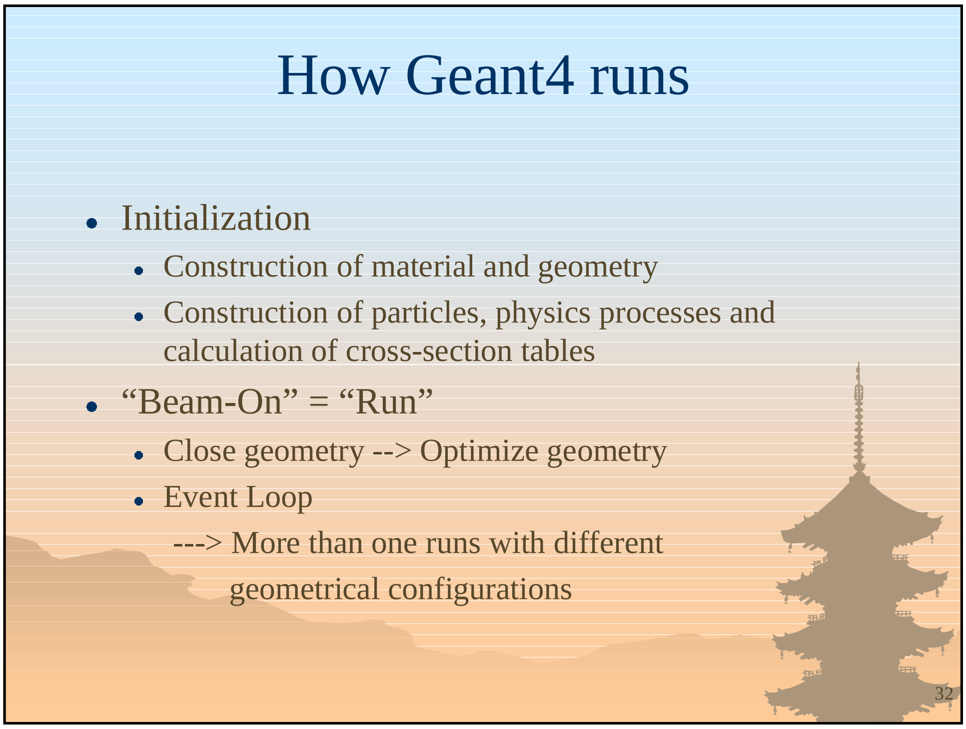
29

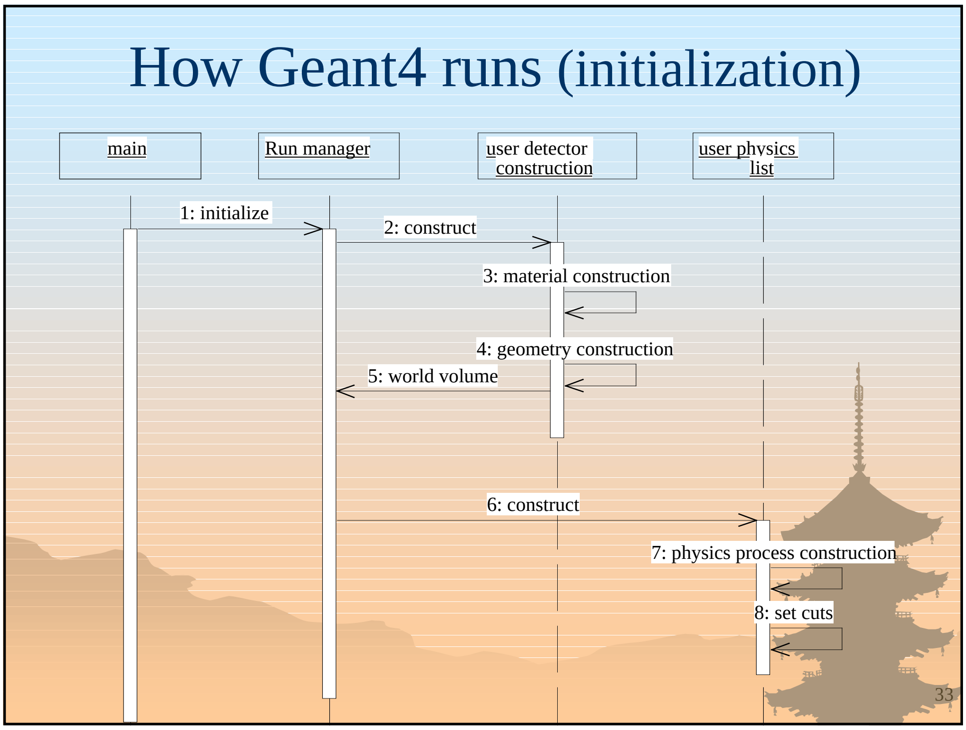
Manager classes

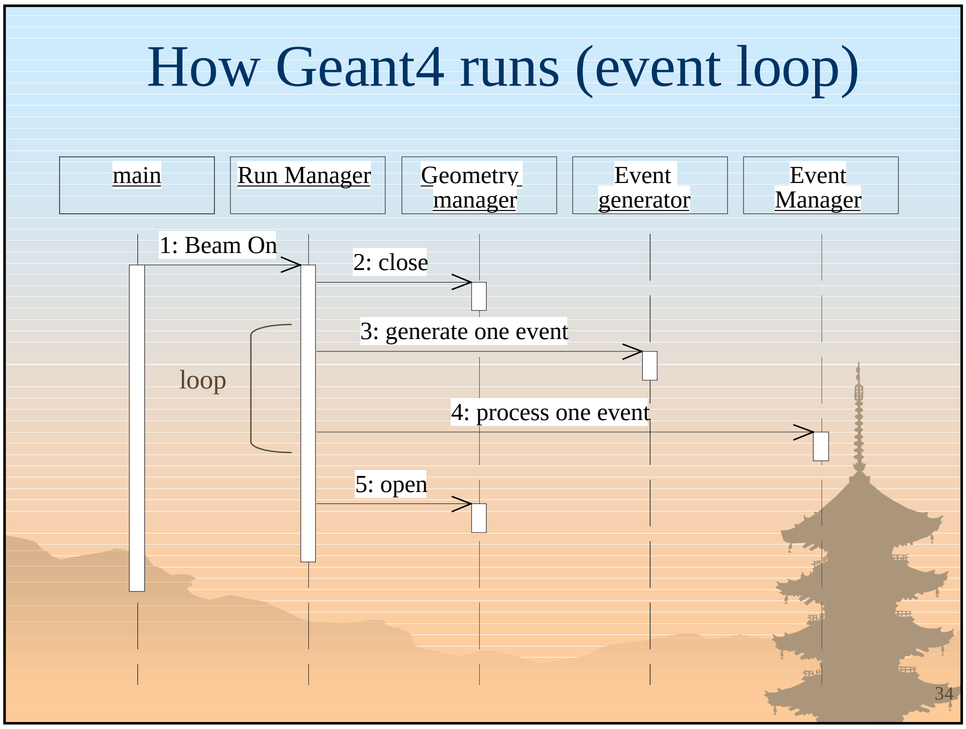
- Geant4 has lots of manager classes (e.g. G4TrackingManager).
  - You may argue that manager classes violate the concept of Object-orientation.
  - But, once a track class has a method "Go_by_yourself()", this class needs to know everything. ---> "Super-class"
  - Having manager classes is our design choice.
    - Localize responsibility
    - Granular categorization

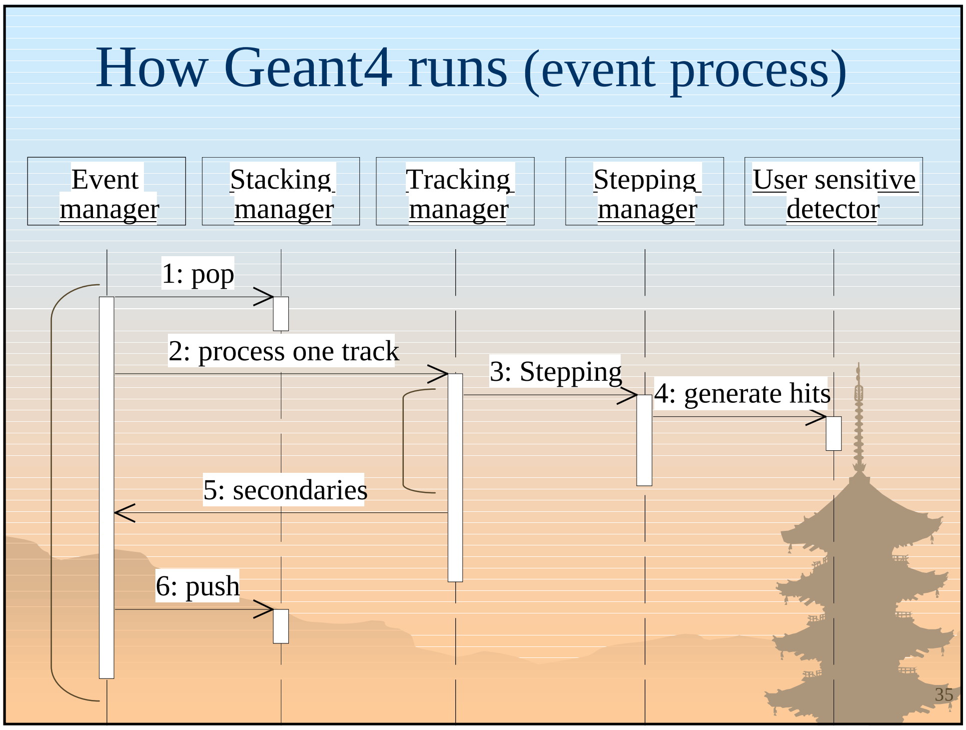
30

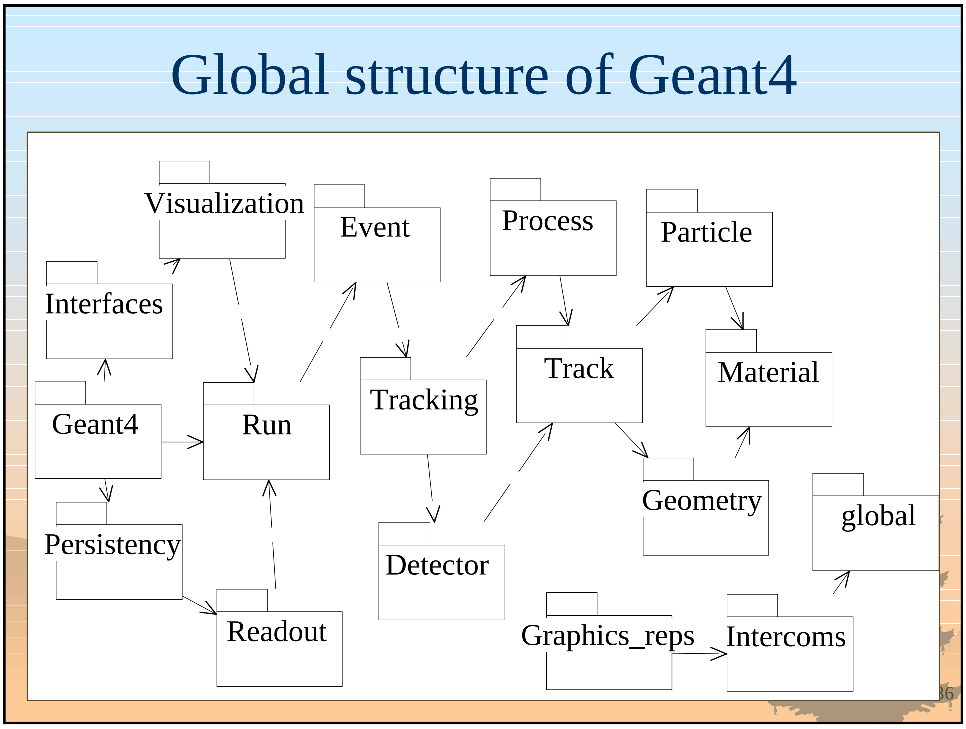

Manager classes
• Manager classes you need to know
• G4RunManager
• G4SDManager
• G4UImanager
• G4FieldManager
• G4VVisManager
Manager classes you'd better to know
•G4EventManager
•G4StackingManager
•G4TrackingManager
•G4SteppingManager
•G4GeometryManager
•G4MatrialManager
•G4VPersistencyManager
(Underline : abstract class)
31


How Geant4 runs
• Initialization
• Construction of material and geometry
• Construction of particles, physics processes and calculation of cross-section tables
• "Beam-On" = "Run"
• Close geometry --> Optimize geometry
• Event Loop
---> More than one runs with different geometrical configurations
32


How Geant4 runs (initialization)
main
Run manager
user detector construction
user physics list
1: initialize
2: construct
3: material construction
4: geometry construction
5: world volume
6: construct
7: physics process construction
8: set cuts
33


How Geant4 runs (event loop)
main
Run Manager
Geometry manager
Event generator
Event Manager
1: Beam On
2: close
3: generate one event
loop
4: process one event
5: open
34


How Geant4 runs (event process)
Event manager
Stacking manager
Tracking manager
Stepping manager
User sensitive detector
1: pop
2: process one track
3: Stepping
4: generate hits
5: secondaries
6: push
35


Global structure of Geant4
Visualization
Event
Process
Particle
Interfaces
Track
Material
Tracking
Geant4
Run
Geometry
global
Persistency
Detector
Readout
Graphics_reps
Intercoms
36

PART 2

The minimal things you have to do

The main program

- Geant4 does not provide the *main*().
- In your *main()*, you have to
  - Construct G4RunManager (or your derived class)
  - Set user mandatory classes to RunManager
    - G4VUserDetectorConstruction
    - G4VUserPhysicsList
    - G4VUserPrimaryGeneratorAction
- You can define VisManager, (G)UI session, optional user action classes, and/or your persistency manager in your *main()*.

38

Describe your detector

- Derive your own concrete class from G4VUserDetectorConstruction abstract base class.
- In the virtual method *Construct*(),
  - Construct all necessary materials
  - Construct volumes of your detector geometry
  - Construct your sensitive detector classes and set them to the detector volumes
- Optionally you can define visualization attributes of your detector elements.

39

Select physics processes

- Geant4 does not have any default particles or processes.
  - Even for the particle transportation, you have to define it explicitly.
- Derive your own concrete class from G4VUserPhysicsList abstract base class.
  - Define all necessary particles
  - Define all necessary processes and assign them to proper particles
  - Define cut-off ranges
- Geant4 provides lots of utility classes/methods.

40

Generate primary event

- Derive your concrete class from G4VUserPrimaryGeneratorAction abstract base class.
- Pass a G4Event object to one or more primary generator concrete class objects which generate primary vertices and primary particles.
- Geant4 provides two generators.
  - G4ParticleGun
  - G4HEPEvtInterface
    - Interface to /hepevt/ common block via ascii file
  - PYTHIA interface will be available quite soon when C++ version of PYTHIA is ready.
  - Interface to HepMC is planned.

41

Environment variables

- You need to set following environment variables to compile, link and run Geant4-based simulation.
  - Mandatory variables
    - G4SYSTEM – OS (e.g. Linux-g++)
    - G4INSTALL – base directory of Geant4
    - G4WORKDIR – your temporary work space
    - CLHEP_BASE_DIR – base directory of CLHEP
  - Variable for physics processes
    - G4LEVELGAMMADATA – directory of PhotonEvaporation data

42

Environment variables

- Variables for GUI and visualization
  - G4UI_USE_TERMINAL
  - G4VIS_USE_OPENGLX
  - G4VIS_USE_DAWN
  - G4VIS_USE_DAWNFILE
  - Variables for DAWN
    - DAWN_HOME – base directory of DAWN
    - DAVID_HOME – base directory of DAVID
    - G4DAWN_NAMED_PIPE
    - DAVID_DAWN_PVNAME
- Installation/configuration scripts will be made available soon

43

PART 3

Add optional features

Select (G)UI

- In your *main*(), according to your computer environments, construct a G4UIsession concrete class provided by Geant4 and invoke its *sessionStart*() method.
- Geant4 provides
  - G4UIterminal -- C-shell like character terminal
  - G4GAG -- Tcl/Tk or Java PVM based GUI
  - G4Wo -- Opacs
  - G4UIBatch -- Batch job with macro file

45

Visualization

- Derive your own concrete class from G4VVisManager according to your computer environments.
- Geant4 provides interfaces to graphics drivers
  - DAWN -- Fukui renderer
  - RayTracer -- Ray tracing by Geant4 tracking
  - OPACS
  - OpenGL
  - OpenInventor
  - VRML

46

Optional user action classes

- All user action classes, methods of which are invoked during “Beam On”, must be constructed in the user’s *main*() and must be set to the RunManager.
- G4UserRunAction
  - BeginOfRunAction(const G4Run*)
    - Define histograms
  - EndOfRunAction(const G4Run*)
    - Store histograms

47

Optional user action classes

- G4UserEventAction
  - BeginOfEventAction(const G4Event*)
    - Event selection
    - Define histograms
  - EndOfEventAction(const G4Event*)
    - Analyze the event

48

Optional user action classes

- G4UserStackingAction
  - PrepareNewEvent()
    - Reset priority control
  - ClassifyNewTrack(const G4Track*)
    - Invoked every time a new track is pushed
    - Classify a new track -- priority control
      - Urgent, Waiting, PostponeToNextEvent, Kill
  - NewStage()
    - Invoked when the Urgent stack becomes empty
    - Change the classification criteria
    - Event filtering (Event abortion)

49

Optional user action classes

- G4UserTrackingAction
  - PreUserTrackingAction(const G4Track*)
    - Decide trajectory should be stored or not
    - Create user-defined trajectory
  - PostUserTrackingAction(const G4Track*)
- G4UserSteppingAction
  - UserSteppingAction(const G4Step*)
    - Kill / suspend / postpone the track
    - Draw the step (for a track not to be stored by a trajectory)

50

PART 4

Basics about geometry

Unit system

- Geant4 has no default unit. To give a number, unit must be "multiplied" to the number.
  - for example :

    double width = 12.5*m;

    double density = 2.7*g/cm3;
  - Almost all commonly used units are available.
  - The user can define new units.
  - Refer to geant4/source/global/management/

      include/SystemOfUnits.h
- Divide a variable by a unit you want to get.

    G4cout << dE / MeV << " (MeV)" << G4endl;

52

Material

- Single element

```
double density = 1.390*g/cm3;
double a = 39.95*g/mole;
G4Material* lAr = new G4Material(name="liquidArgon",
   z=18., a, density);
```

- There must be no vacuum.
  - Use very low density instead.

53

Material

- Molecule

```
a = 1.01*g/mole;
G4Element* elH  = new G4Element(name="Hydrogen",
   symbol="H" , z= 1., a);
a = 16.00*g/mole;
G4Element* elO  = new G4Element(name="Oxygen",
   symbol="O" , z= 8., a);
density = 1.000*g/cm3;
G4Material* H2O = new G4Material(name="Water",
   density, ncomponents=2);
H2O->AddElement(elH, natoms=2);
H2O->AddElement(elO, natoms=1);
```

54

Material

- Compound

a = 14.01*g/mole;

G4Element* elN = new G4Element(name="Nitrogen", symbol="N" , z= 7., a);

a = 16.00*g/mole;

G4Element* elO = new G4Element(name="Oxygen", symbol="O" , z= 8., a);

density = 1.290*mg/cm3;

G4Material* Air = new G4Material(name="Air", density,ncomponents=2);

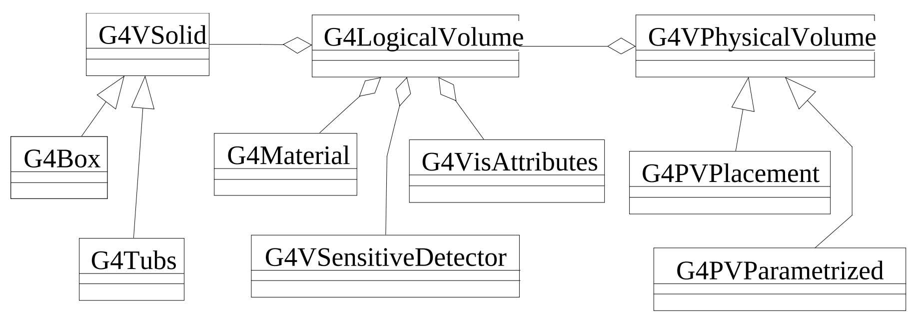
Air->AddElement(elN, 70.0*perCent);

Air->AddElement(elO, 30.0*perCent);

55

Define detector geometry

- Three conceptual layers
  - G4VSolid -- shape, size
  - G4LogicalVolume -- daughter phys. volumes, material, sensitivity, user limits, etc.
  - G4VPhysicalVolume -- position, rotation



Define detector geometry

- Basic strategy

```
G4VSolid* pBoxSolid = new G4Box("aBoxSolid",
        1.*m, 2.*m, 3.*m);
G4LogicalVolume* pBoxLog = new G4LogicalVolume(
      pBoxSolid, pBoxMaterial, "aBoxLog", 0, 0, 0);
G4VPhysicalVolume* aBoxPhys = new G4PVPlacement(
      pRotation, G4ThreeVector(posX, posY, posZ),
      pBoxLog, "aBoxPhys", pMotherLog, 0, copyNo);
```

- A unique physical volume which represents the experimental area must exist and it fully contains all of other components.

  ---> The world volume

57

Define detector geometry

- G4VSolid
  - CSG solids
    - G4Box, G4Tubs, G4Cons, G4Trd, etc.
    - Analogy to GEANT3 solids
  - BREP solids
    - G4BREPSolid, G4BSplineSurface, etc.
  - Boolean solids
    - G4UnionSolid, G4SubtractionSolid, etc.
  - STEP interface
  - SWEPT solids are planned.

58

Define detector geometry

- G4LogicalVolume
  - Contains all information of volume except position:
    - Shape and dimension (G4VSolid)
    - Material, sensitivity, visualization attributes
    - Position of daughter volumes
    - Magnetic field, User limits
    - Shower parameterization
  - Physical volumes of same type can share a logical volume.
  - It has several basic Set methods.

59

Define detector geometry

- G4VPhysicalVolume
  - G4PVPlacement 1 Placement = One Volume
    - Simple placement
  - G4PVParameterized 1 Parameterized = Many Volumes
    - Reduction of memory consumption
    - Parameterized by the copy number
    - Shape, size, material, position and rotation can be parameterized
      - by implementing a concrete class of G4VPVParameterisation.
    - Currently, parameterization must be applied only for the "leaf" volumes.
  - G4PVReplica 1 Replica = Many Volumes
    - Slicing a volume into smaller pieces (if it has a symmetry)

60

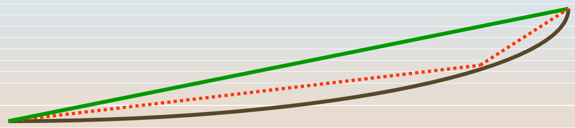
Magnetic field

- In order to propagate a particle inside a field (e.g. magnetic, electric or both), we integrate the equation of motion of the particle in the field.
- In general this is best done using a Runge-Kutta method for the integration of ordinary differential equations. Several Runge-Kutta methods are available.
- In specific cases other solvers can also be used:
  - In a uniform field as the analytical solution is known.
  - In a nearly uniform field where we perturb it.

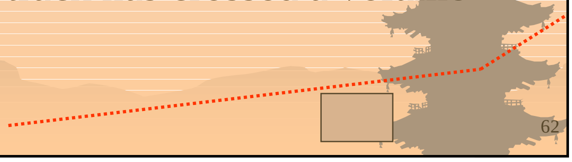
61

Magnetic field

- Once a method is chosen that allows G4 to calculate the track's motion in a field, we break up this curved path into linear chord segments.



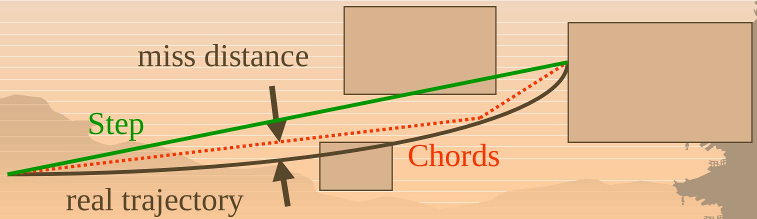
- We determine the chord segments so that they closely approximate the curved path.
- We use the chords to interrogate the Navigator, to see whether the track has crossed a volume boundary.



62

Magnetic field

- You can set the accuracy of the volume intersection,
  - by setting a parameter called the “miss distance”
    - it is a measure of the error in whether the approximate track intersects a volume.
    - Default “miss distance” is 3 mm.
- One step can consist of more than one chords.
  - In some cases, one step consists of several turns.



63

Magnetic field

- Magnetic field class
  - Uniform field :

    G4UniformMagField class object
  - Non-uniform field :

    Concrete class derived from G4MagneticField
- Set it to G4FieldManager and create a Chord Finder.

```
G4FieldManager* fieldMgr
    = G4TransportationManager::GetTransportationManager()
      ->GetFieldManager();
fieldMgr->SetDetectorField(magField);
fieldMgr->CreateChordFinder(magField);
```

64

Touchable

- As mentioned already, G4Step has two G4StepPoint objects as its starting and ending points. All the geometrical information of the particular step should be got from “PreStepPoint”.
  - Geometrical information associated with G4Track is basically same as “PostStepPoint”.
- Each G4StepPoint object has
  - Position in world coordinate system
  - Global and local time
  - Material
  - G4TouchableHistory for geometrical information

65

Touchable

- G4TouchableHistory has information of geometrical hierarchy of the point.

```
G4Step* aStep;
G4StepPoint* preStepPoint = aStep->GetPreStepPoint();
G4TouchableHistory* theTouchable
   = (G4TouchableHistory*)(preStepPoint->GetTouchable());
G4int copyNo = theTouchable->GetVolume()->GetCopyNo();
G4int motherCopyNo = theTouchable->GetVolume(1)->GetCopyNo();
G4ThreeVector worldPos = preStepPoint->GetPosition();
G4ThreeVector localPos
   = theTouchable->GetHistory()->GetTopTransform().TransformPoint(worldPos);
```

66

PART 5

Sensitive detector and digitizer

Detector sensitivity

- A logical volume becomes sensitive if it has a pointer to a concrete class derived from G4VSensitiveDetector.
- A sensitive detector constructs one or more hit objects or accumulate values to existing hits using information given in a G4Step object.
- Remember to get the volume information from “PreStepPoint”.

68

Hit class

- Hit is a user-defined class derived from G4VHit. You can store various information by implementing your own concrete Hit class.
  - Position and time of the step
  - Momentum and energy of the track
  - Energy deposition of the step
  - Geometrical information
  - or any combination of above

69

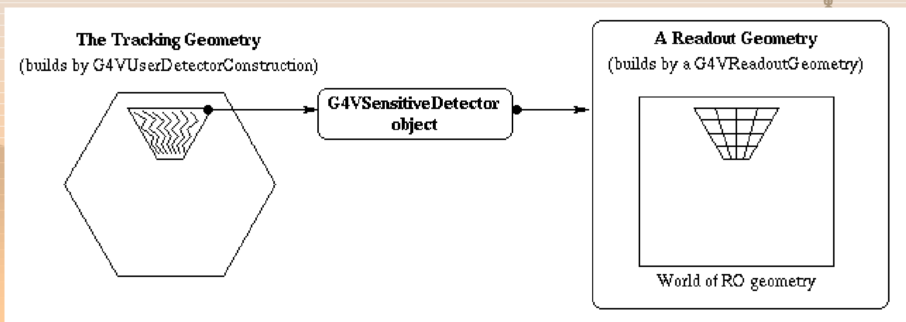
Hit class

- Hit objects of a concrete hit class must be stored in a dedicated collection which is instantiated from G4THitsCollection template class.
- The collection will be associated to a G4Event object via G4HCofThisEvent.
- Hits collections are accessible
  - through G4Event at the end of event,
  - through G4SDManager during processing an event.
    --> Event filtering by StackingAction.

70

Readout geometry

- Readout geometry is a virtual and artificial geometry which can be defined in parallel to the real detector geometry.
- A readout geometry is associated to a sensitive detector.



71

Digitization

- Digit represents a detector output (e.g. ADC/TDC count, trigger signal).
- Digit is created with one or more hits and/or other digits by a concrete implementation derived from G4VDigitizerModule.
- In contradiction to the Hit which is generated at tracking time automatically, the digitize() method of each G4VDigitizerModule must be explicitly invoked by the user’s code (e.g. EventAction).

72

PART 6

Visualization and (G)UI

Visualization of Detector

- Each logical volume can have a G4VisAttributes object.
  - Visibility, visibility of daughter volumes
  - Color, line style, line width
  - Force flag to wire frame mode
- For the parameterized volume case, attributes can be dynamically assigned to the logical volume.

74

Visualization of Hits and Trajectories

- Each G4VHit concrete class must have an implementation of *Draw()* method.
  - Colored marker
  - Colored solid
  - Change the color of detector element
- G4Trajectory class has a *Draw()* method.
  - Blue : positive
  - Green : neutral
  - Red : negative
  - You can implement alternatives by yourself

75

What is Intercoms?

- Intercoms category is used by almost all other Geant4 categories for exchanging information without having pointers.
  - E.g. the user can apply "abort event" command from user stepping action without knowing the pointer to G4EventManager.
  - (G)UI also accepts commands dynamically.
- G4UImanager receives the application of a command and passes it to a messenger. The messenger brings the command to the target destination class object.

76

Command submission

- To submit a command from your code

  G4UImanager* UI = G4UImanager::GetUIpointer();

  UI->ApplyCommand("*full_path_command parameter(s)*");
- Some useful commands
  - /run/beamOn *nEvent*
  - /run/verbose *nLevel*
  - /event/verbose *nLevel*
  - /tracking/verbose *nLevel*
  - /tracking/storeTrajectory *bool*
  - /control/execute *macro_file*

77

Define user commands

- Create a messenger concrete class derived from G4UImessenger and associate it to your target class.
- Construct G4UIcommand or its derived class object to define a command.
- Implement GetCurrentValues() and SetNewValues() method.

```
G4UIcommand* energyCmd = new
  G4UIcmdWithADoubleAndUnit("/gun/energy",this);
energyCmd->SetGuidance("Set kinetic energy.");
energyCmd->SetParameterName("Energy",true,true);
energyCmd->SetRange("Energy > 0.");
energyCmd->SetDefaultUnit("GeV");
```

78

G4cout / G4cerr / G4endl

- G4cout, G4cerr and G4endl are iostream objects defined by Geant4. The user is recommended to use them instead of ordinary cout/cerr/endl. Don’t forget to include “G4ios.hh”.
- GUI manipulates output stream to store logs.
- G4cout/G4cerr should not be used in the constructor of a class if the instance of this class is intended to be used as "static". This restriction comes from the language specification of C++.
- “cin” should not be used. Use intercoms.

79

PART 7

Learn More

User’s manuals

- Introduction to Geant4
- User’s Guide
  - Installation Guide
  - For Application Developer
  - For Toolkit Developer
  - Physics Reference Manual
  - Software Reference Manual
- Contributions from Users
  - Useful samples, notes, FAQs from the users
- Visit http://cern.ch/geant4/

81

Examples

- Novice level examples
  - ExampleN01
    - Demonstrates how Geant4 kernel works
  - ExampleN02
    - Simplified tracker geometry with magnetic field
    - Electromagnetic processes
  - ExampleN03
    - Simplified calorimeter geometry
    - Various materials

82

Examples

- Novice level examples
  - ExampleN04
    - Simplified collider detector with readout geometry
    - EM + Hadronic processes
    - PYTHIA interface
    - Event filtering by stack mechanism
  - ExampleN05
    - Simplified BaBar calorimeter
    - Shower parameterization
  - ExampleN06
    - Optical photon processes

83

Examples

- Extended level examples
  - Persistency by Objectivity/DB (CERN RD45)
  - “G3toG4” conversion tool
  - EM processes for various use-cases
- Advanced level examples
  - To be prepared
  - Expect user’s contributions

84